A Topology-Based Mass/Spring System

Philippe Meseure, Emmanuelle Darles, Xavier Skapin

A Topology-Based Mass/Spring System *

Philippe Meseure, Emmanuelle Darles, Xavier Skapin
Philippe.Meseure@sic.univ-poitiers.fr
University of Poitiers, XLIM-SIC CNRS 6172

Abstract

This paper presents how to design and implement a simple mass/spring system on a general-purpose topological structure. This way, topological inquiries or changes can be handled robustly so that the mesh remains consistent and a manifold.

Keywords: physically-based animation, generalized maps, surgical simulation

Introduction

Among the various physically-based models for animation, most of them need to include some topological information that is usually partial and dedicated to specific needs. However, in different domains, the simulated body may undergo topology changes whose consistency cannot be fully guaranteed using a naive topological model. Examples include fractures, fusion and, in surgical simulation, cuttings or tearings. These two last transformations can be handled in two ways: (a) dissociation of volumes [1, 2] and (b) removal of volumes [3]. A third approach has been proposed these last years and consists in duplicating volumes [4]. All these methods heavily rely on adjacency relations in the mesh. [5] showed that the simulated body should remain a *manifold* during the topological changes, in order to avoid ill-structured elements. Without a topological model, guaranteeing the manifold structure leads to complex and heavy treatments of the mesh. Some studies have tried to use specific topological models [6, 7] but are restricted to dimension 2.

We aim at providing physical models with a complete and versatile topological base which guarantees robustness, versatility and avoids ill-structured objects. The paper is organized as follows. After describing the generic topological model we used, we present how a mass/spring system can be based on a topological model and, in the following section, the topological modifications we have applied. Section 4 gives our results and we conclude.

1 Topological models

We rely on *generalized maps* [8], since they are equivalent to most of other $3D$ topological models [9]. They allow to represent orientable or non-orientable quasi-manifolds. Their algebraic definition is homogeneous (in dimension n) while their data structure is simple and intuitive to manipulate.

In the remainder of the paper, we will refer to vertices, edges, faces and volumes respectively as 0-cells, 1-cells, 2-cells and 3-cells. An n-dimensional generalized map (denoted n-g-map) is based on a unique abstract element, called *dart*, that, for $3D$-objects, roughly represents "a vertex on an edge of a face of a volume of the object". n-g-maps are defined as a set of darts and applications α_i defined on these darts that allow to represent the various adjacency relationships. Each α_i aims at "sewing" the i-cell along a $(i-1)$-cell (for instance, sewing two faces by sharing a common edge). The image by α_i is defined for every dart of the sewed entity. A formal definition can be found in [8]. An

*Proceedings of Computer Animation and Social Agents, St Malo (France), 31 may-2 june 2010.

"orbit" is the set of all darts that can be reached starting from a given dart b, using a combination of selected involutions. An i-cell appears as the orbit built using all involutions except α_i.

The topological model only describes the structure of the object. The geometric information, such as the positions of the vertices, must be added. For that purpose, it is possible to embed information in the structure. An embedded information can be attached to any i-cell. In practice, this means that all the darts of the given i-cell share the same information. Any type of information can be embedded: geometric, visual or even mechanical in our context.

2 A topology-based spring system

2.1 Embeddings

To design our model, the first task consists in embedding the topological model with mechanical information. Without loss of generality, we suppose here that the mesh is only composed of tetrahedra. In a tetrahedron, we embed its mass. In an edge, we embed the rest-length of the corresponding spring, the stiffness and, if needed, a damping value. We select the outside faces of the object and embed into them some details required by the rendering processes, such as the normal for instance. Finally, in a vertex, we embed the mass of the node (obtained by summing the contribution of all the surrounding tetrahedra), an ambient viscosity coefficient and the position and velocity of the node. It is wise, for optimization purposes, to also include the sum of all the forces applied to the node. Note that the structure of the mechanical mesh must be quasi-manifold and that each spring is mapped to an edge of the mesh.

2.2 Behavior computation

Since the different mechanical properties and state variables are embedded into the structure, all the usual algorithms must be designed accordingly. Indeed, neither the nodes nor the springs appear as arrays that we only have to parse. On the contrary, each algorithm must rely on an adapted coverage of the topological structure.

To cover all the i-cell for a given i, we basically check every dart, treat the corresponding i-cell and mark all the darts that belong to the same i-cell orbit. While parsing the structure, only the unmarked darts are taken into account before they are marked. At the end, all the darts are marked. At this stage, a global coverage of the structure aims at unmarking the darts.

To compute the deformation forces, we cover all the edges of the structure. Let b a dart that belongs to an edge orbit. We can easily get the extremity nodes of the spring: b belongs to the vertex orbit of one of the nodes, and $\alpha_0(b)$ to the other one. We can then compute the deformation force relying on the position and velocity of the extremity nodes and apply this force to them.

The force integration and state variables computation require to cover all the vertices in a similar way. The process is rather straightforward for explicit integration schemes such as Euler or Runge Kutta. However, more elaborate integration methods rely on linear algebra. For such methods, it is recommended to associate each vertex with an index in the state variables vector. A final walk through the structure allows each vertex to pick up its new position and velocity in the state vector.

Finally, because the collision process can be handled in different ways, we cannot make a complete inventory of all the possibilities. Depending on the chosen approach, the vertices, the faces or the volumes must be parsed. Anyway, the topological model always allows us to find the concerned vertices, get their position and velocity to compute the collision correctly and apply the resulting forces to them.

3 Topological Operations

Among the topological changes used in surgical simulation, let us consider the volume separation, namely "face split" first. Using g-maps, this operation only corresponds to the α_3-unsew operation between the two volumes and is really straightforward: The algorithm walks through the darts of the face (exploring α_0 and α_1 links), and discards the α_3 links. During this process, the cancellation of α_3 may induce vertex and/or edge split, as shown in figure 1 in the 2D case. This can be easily detected, since in this case,

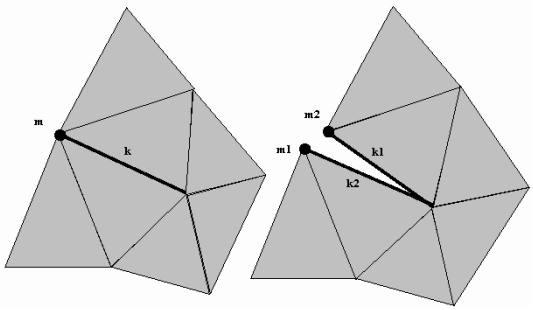


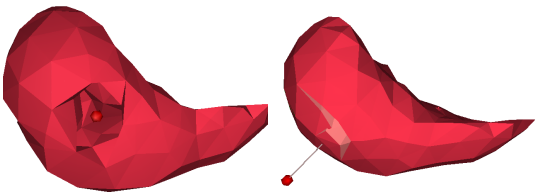
Figure 1: Mechanical adaptation after the splitting of a vertex and an edge.

two unsewed darts no longer belong to the same vertex/edge orbit. This is checked when the face split is over.

When a vertex is split, we must prevent the two resulting vertices from sharing the same embedding. We therefore duplicate the position and velocity but have to compute the new masses. The mass of a vertex is obtained by summing up the mass contribution of all the adjacent tetrahedra (for the sake of simplicity, we suppose that a tetrahedron provides each of its vertex with $1/4$ of its mass). This way, we ensure that the sum of the masses of the two vertices is always the mass of the original split vertex. We then consider that the ambient viscosity is proportional to the computed mass. Note that if the vertices have an index toward a state vector, we have to supply one of the appearing vertices with a new index (the other vertex keeps the index of the split vertex).

The edge split process is rather similar. If two unlinked darts no longer belong to the same edge orbit, an edge split has occurred. We then consider all the adjacent tetrahedra of the two resulting edges and sum up all their contribution to obtain the resulting stiffness. This way, the sum of the stiffness of the two edges is the same as the stiffness of the initial split edge, as stated by the Kirchhoff law applied to parallel combination of springs.

The other operation we have considered is volume deletion. A straightforward but naive approach consists in erasing all the darts of the volume. However, it can be quite hazardous to control embeddings correctly using such an approach. When a dart is deleted, all its embeddings are dispatched on other darts belonging to the initial i-cell of the dart. However, as seen



(a) (b)

Figure 2: Tetrahedron removals and face splits.

previously, the mechanical embeddings should be adapted according to the destruction, a task that becomes tricky because of the loss of information implied by the deletion of the darts. It is more convenient to isolate the volume first, by iterating on all its faces and applying the previously-defined face split. Such a technique guarantees to share masses, stiffnesses and so on where needed. When the volume is isolated, all its darts and corresponding embeddings can be deleted. Note that the unsewed faces of the tetrahedra surrounding the removed tetrahedron become 3-free and must be taken into account in the rendering step (since they belong to the boundary of the object).

4 Results

In our implementation, we have used the MOKA library (freely available at *http://moka-modeller.sourceforge.net*) to handle g-maps. The mechanics and the interaction tools are handled within a home-made dynamic simulator [10], using penalty methods between the tetrahedra to handle collision. The figure 2 shows result of an object (a liver) after several tetrahedron removals. Using a dual core 2.8GHz processor, we need 8 ms to compute an evolution step of a deformable body including 564 nodes, 2850 springs and 1856 tetrahedra (using Runge Kutta 4).

Note that various optimizations are required. Marking only a subset of the darts while walking through the structure should be avoided since this subset has to be covered once more to erase the marks. We have used a property of the MOKA library which consists in embedding information of an orbit only in a single dart of the orbit and provide the other darts with shortcuts to this information. Thus, to cover all i-cells,

it is sufficient to check all the darts and only consider the ones that include the i-cells embedding. This way, no marker is required, and most implemented walks through the structure have a linear complexity *w.r.t.* the number of darts. We also include all the node embeddings in a list to speed up the loops that use these information without requiring any adjacency relationships.

We have compared the computation time of our model to the one of a mass/spring model only based on indexed structures (with no topology change). The indexed mass/spring body needs 2ms for a time step. In other words, our topologically-based model is four times as slow as the indexed model. This is mainly due to the number of darts that need to be covered at each step of the algorithm, in our example 44544 darts (each tetrahedron requires 24 darts).

Conclusion and Future work

In this paper, we have presented how to base a mass/spring system on a generalized-map model. The computation of neighborhood properties is simplified as well as the design of operations. We have also shown how to compute the new embeddings when some topological change is applied. We find that our model is four times as slow as a naive indexed approach. It is however possible to get better results. In particular, a straightforward implementation of g-maps heavily relies on pointers and leads to a fragmentation of the memory that an optimized implementation would avoid. In the future, we also intend to implement topology inquiries (checking the connexity of the model) and more complex operations.

Acknowledgment

This work has been funded by the French Research Agency through the VORTISS Project (ANR-06-MDCA-015). We also want to thank the IRCAD (Strasbourg) for the liver model and G. Damiand for helping us to use MOKA.

References

[1] S. Frisken-Gibson. Using linked volumes to model object collisions, deformation, cutting, carving and joining. *IEEE Trans. on Visual. and Comp. Graph.*, 5(4):333–348, 1999.

[2] H.W. Nienhuys and A.F. van der Stappen. Combining finite element deformation with cutting for surgery simulations. In *EUROGRAPHICS'00*, pages 43–51, 2000.

[3] S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model allowing real-time cutting, deformation and force-feedback for surgery training and simulation. *The Visual Comp.*, 16(8):437–452, 2000.

[4] N. Molino, Z. Bao, and R. Fedkiw. A virtual node algorithm for changing mesh topology during simulation. In *ACM Trans. on Graph. (SIGGRAPH)*, volume 23, pages 385–392, 2004.

[5] C. Forest, H. Delingette, and N. Ayache. Removing tetrahedra from manifold tetrahedralisation: application to real-time surgical simulation. *Med. Im. Analysis*, 9(2):113–122, April 2005.

[6] B. G. Baumgart. A polyhedron representation for computer vision. In *AFIPS'75*, pages 589–596, 1975.

[7] H. Delingette, G. Subsol, S. Cotin, and J. Pignon. A craniofacial surgery simulation testbed. In *Visual. and Biomed. Comp.*, pages 607–18, 1994.

[8] P. Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *Int. J. of Comp. Geom. and App.*, 4(3):275–324, 1994.

[9] P. Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer Aided Design*, 23:59–82, 1991.

[10] P. Meseure et al. A physically-based virtual environment dedicated to surgical simulation. In *Int. Symp. on Surg. Sim. and*

Soft Tissue Mod., volume 2673 of *LNCS*, pages 38–47, June 2003.